Lecture 18: EM

Fall 2022

Kai-Wei Chang
CS @ UCLA
kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Final Exam

- Official exam time 12/9 11:30am -- 2:30pm
- Closed-book in-person exam at Moore Hall 100
- We expect to complete the grading in about a week
- The final grade will be computed based on the formulation in Lecture 1

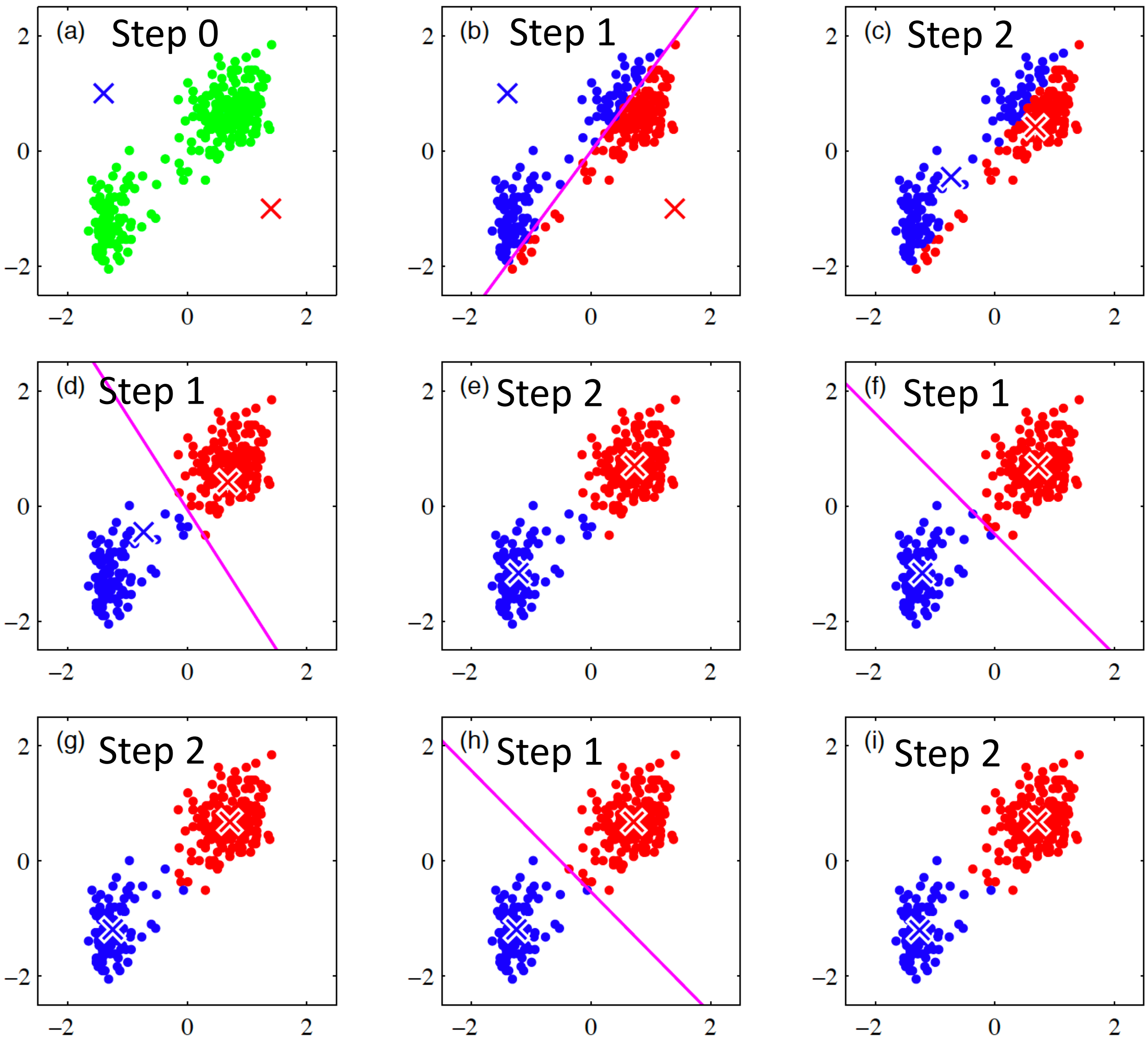
- Cover everything until today's lecture

Course evaluation survey

- Please complete the course survey at MyUCLA by 12/2
  - Your feedback is important for us as we continue improving the courses and incorporating new materials
  - 2 bonus points for the final for students filling out the survey
    - Your response is anonymous

Unsupervised Learning

Recap: K-Means



Recap: K-means algorithm

- Step 0: randomly assign the cluster centers $\{\mu_k\}$
- Step 1: Minimize J over $\{r_{nk}\}$ -- Assign every point to the closest cluster center

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- Step 2: Minimize J over $\{\mu_k\}$ -- update the cluster centers

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

- Loop until it converges

Recap: K-means algorithm

- Step 0: randomly assign the cluster centers $\{\mu_k\}$
- Step 1: Minimize J over $\{r_{nk}\}$ -- Assign every point to the closest cluster center

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

Assign data to clusters based on the model (E-step)

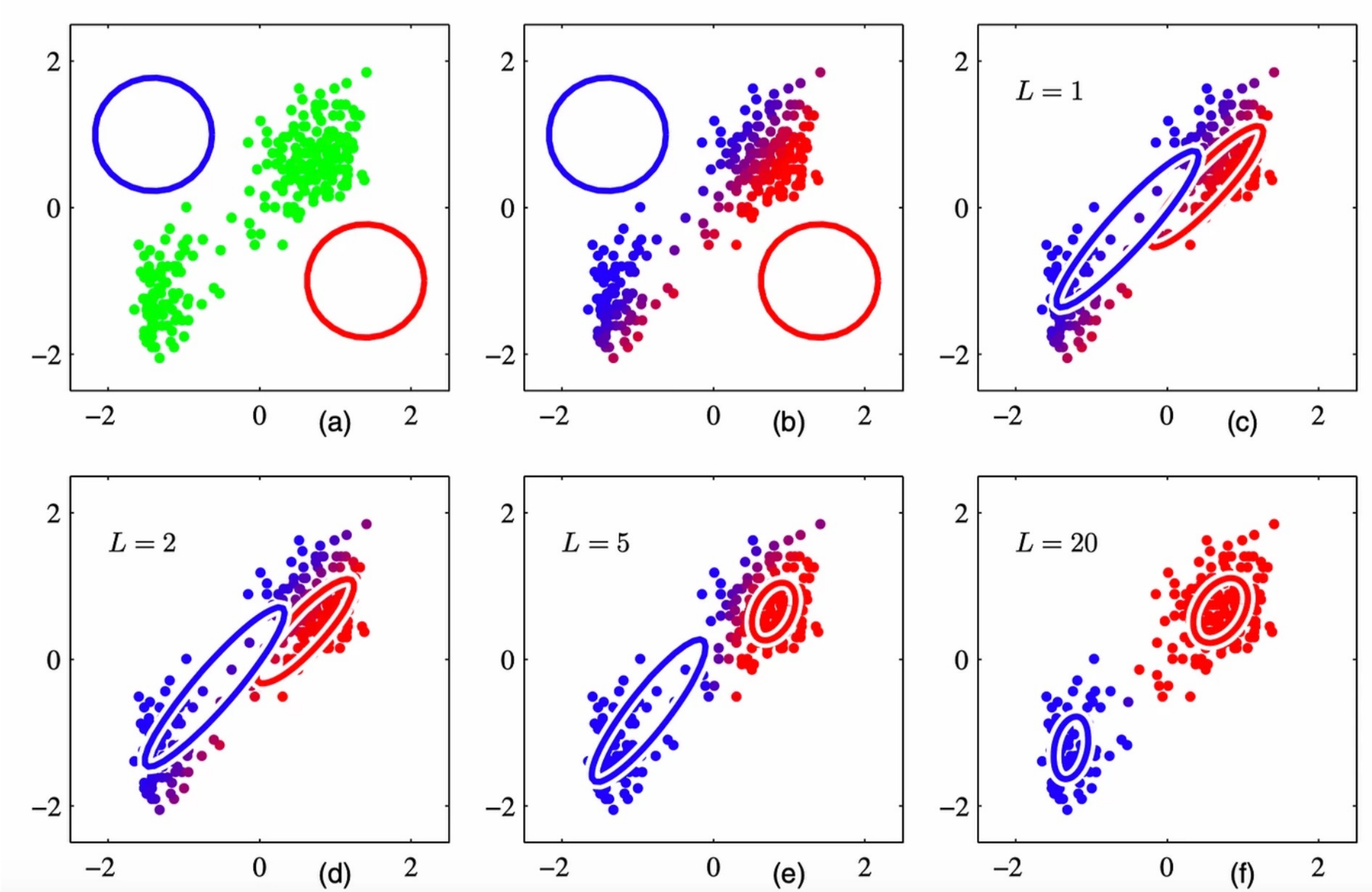
- Step 2: Minimize J over $\{\mu_k\}$ -- update the cluster centers

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

Updating the model (M-Step)

- Loop until it converges

Recap: GMM



Assume each cluster can be modeled by a Gaussian with cluster center μ_k and covariance matrix Σ_k i.e., P(x|cluster=k) = $\mathrm{N}(x|\mu_k, \Sigma_k)$

From K-means to GMM

We will go back to explain formulation later

- Step 0: randomly assign the cluster centers $\{\mu_k\}$ and covariance matrices $\{\Sigma_k\}$
- Step 1: Assign every point to the cluster based on posterior distribution $P(z_n = k|x_n)$

$$\boxed{p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}}$$

- Step 2: update the cluster centers $\{\mu_k\}$, and covariance matrices $\{\Sigma_k\}$ based on soft assignment $\gamma_{nk} = P(z_n = k|x_n)$

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

- Loop until it converges

γ_{nk} is similar to r_{nk} in K-means, but $\gamma_{nk} \in (0,1), r_{nk} \in \{0,1\}$

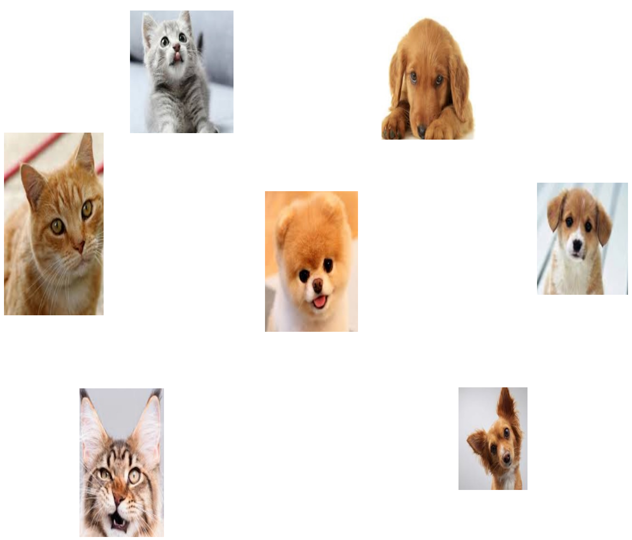
Unsupervised Learning – Expectation Maximization (EM)

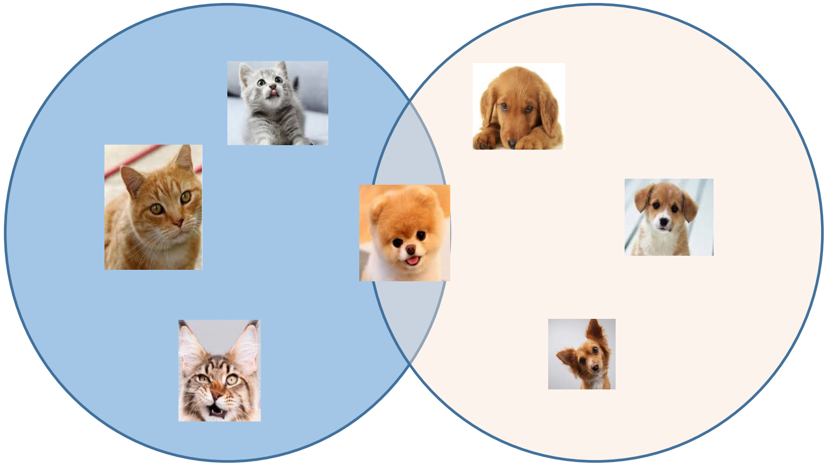
Roadmap

- The goal of EM algorithm
- A numerical example
- GMM (instance of EM)

How about unsupervised learning

- In unsupervised learning, we only observed input distribution $P(X)$





MLE in unsupervised learning

- We only have the observation of $\mathrm{P}(X)$
- We make assumptions to model $P(X, Y \mid \Theta)$
- We know $P(X \mid \Theta) = \sum_Y P(X, Y \mid \Theta)$
- Therefore, MLE is

$$
\begin{aligned}
&\operatorname{argmax}_\Theta P(X \mid \Theta) \\
&= \operatorname{argmax}_\Theta \sum_Y P(X, Y \mid \Theta) \\
&= \operatorname{argmax}_\Theta \log \sum_Y P(X, Y \mid \Theta)
\end{aligned}
$$

EM algorithm

- EM algorithm essentially solves $\mathrm{argmax}_{\Theta} \log \sum_Y P(X, Y \mid \Theta)$ by iteratively updating Θ (come back to this point later)
  - At iteration t, the model Θ^t
  - E-Step: Estimate P(Y | X, Θ^t) Step 1 in GMM
  - M-Step: Optimize

    $\max_{\Theta} \sum_{\mathrm{Y}}[P(\mathrm{Y} \mid \mathrm{X}, \Theta^t) \log \mathrm{P}(\mathrm{X}, \mathrm{Y} | \Theta)]$ Step 2 in GMM
- In general, it converges to a local maximum

Roadmap

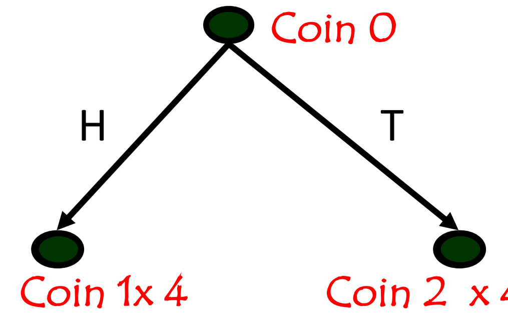
- The goal of EM algorithm
- A numerical example
- GMM (instance of EM)

Three Coins Example

- We observe a series of coin tosses generated in the following way:
- A person has three coins.
  - Coin 0: probability of Head is α
  - Coin 1: probability of Head p
  - Coin 2: probability of Head q

- Consider the following coin-tossing scenarios:

Scenario I



- Toss coin 0
  If Head – toss coin 1 x 4 times;
  o/w – toss coin 2 x 4 times

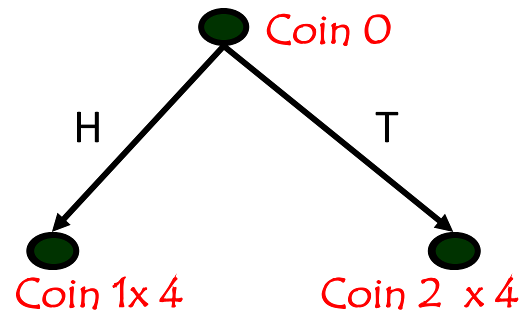
  Observing the sequence
  HHHHT, THTHT, HHHHT, HHTTH

  produced by Coin 0 , Coin1 and Coin2

  Question: Estimate most likely values for α, p, q (the probability of H in each coin) and the probability to use each of the coins

Supervised Learning

Scenario I



- Toss coin 0
  If Head – toss coin 1 x 4 times;
  o/w – toss coin 2 x 4 times

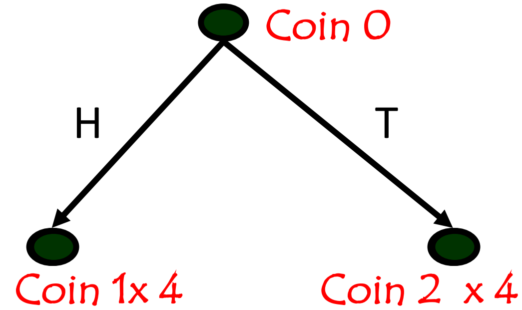
- Observing the sequence
  HHHHT, THTHT, HHHHT, HHTTH

  produced by Coin 0 , Coin1 and Coin2

  Question: Estimate most likely values for α, p, q (the probability of H in each coin) and the probability to use each of the coins

$\alpha=3/4$, p=8/12, q=2/4

Scenario II



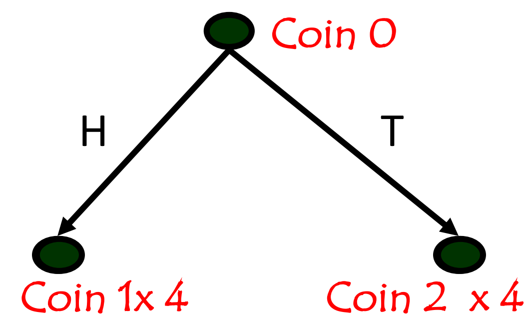
- Toss coin 0
  If Head – toss coin 1 x 4 times;
  o/w – toss coin 2 x 4 times

Now we only observe outcomes from coins 1 & 2
?HHHT, ?HTHT, ?HHHT, ?HTTH

Question: Estimate most likely values for α, p, q (the probability of H in each coin) and the probability to use each of the coins

Guess Coin 0 based on the model

- Toss coin 0
  If Head – toss coin 1 x 4 times;
  o/w – toss coin 2 x 4 times



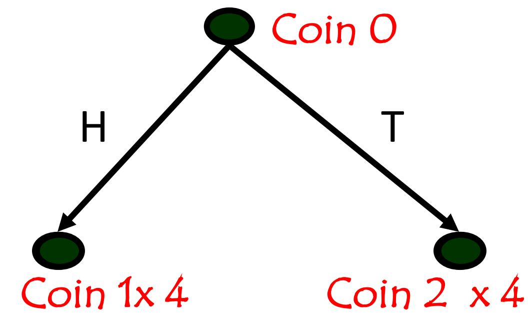
Assume $\alpha=3/4$, p=2/3, q=1/2

If we observe the outcome sequence is
?HHHT

Question: How likely the sequence is from Coin 1 vs Coin 2

hint: Bayes' theorem

Guess Coin 0 based on the model



- Toss coin 0
  If Head – toss coin 1 x 4 times;
  o/w – toss coin 2 x 4 times

Assume $\alpha=3/4$, p=2/3, q=1/2

Question: How likely the sequence ?HHHT is from Coin 1 vs Coin 2

$$P(HHHHT) = \frac{3}{4} \times \left(\frac{2}{3}\right)^3 \times \left(\frac{1}{3}\right)$$

$$P(THHHT) = \frac{1}{4} \times \left(\frac{1}{2}\right)^3 \times \left(\frac{1}{2}\right)$$

$$P(coin0 = H \mid ?HHHT) = \frac{P(HHHHT)}{P(HHHHT) + P(THHHT)}$$

Intuition of EM algorithm

- Use an iterative approach for estimating the parameters:
  - Guess the probability that a given data point came from Coin 1 or 2; Generate fictional labels, weighted according to this probability.
  - Then, compute the most likely value of the parameters. [supervised learning]
  - Compute the likelihood of the data given this model.
  - Re-estimate the parameter setting: set them to maximize the likelihood of the data.

Step 1: Random initialization

- Guess the probability that a given data point came from Coin 1 or 2 randomly

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

Sequence	coin 0=H	coin 0=T
HHHT	100%	0 %
HTHT	100%	0%
HHHT	100%	0%
HTTH	0%	100%

Step 2: Maximum Conditional Likelihood

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

- Now, compute the most likely value of the parameters. [Supervised Learning]

Sequence	coin 0=H	coin 0=T
HHHT	100%	0 %
HTHT	100%	0%
HHHT	100%	0%
HTTH	0%	100%

HHHHT
HHTHT
HHHHT
THTTH

E-STEP - guess labels

Step 2: Maximum Conditional Likelihood

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

- Now, compute the most likely value of the parameters.

Sequence

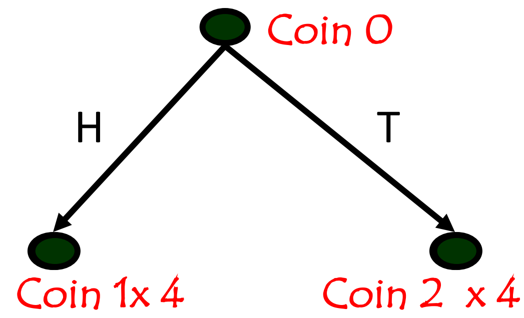
HHHHT
HHTHT
HHHHT
THTTH

$$\alpha_1 = \frac{3}{3+1} = \frac{3}{4}$$

$$p_1 = \frac{8}{8+4} = \frac{2}{3}$$

$$q_1 = \frac{2}{2+2} = \frac{1}{2}$$

Step3: Compute Posterior



$$P(coin\ 0 \mid Sequence; \alpha_1, p_1, q_1)$$

- Compute the likelihood of the data given this model

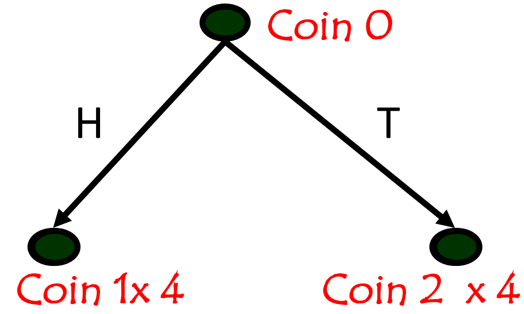
Sequence	$\alpha_1 p_1^{\#H}(1-p_1)^{\#T}$ coin 0 =H	$(1-\alpha_1) q_1^{\#H}(1-q_1)^{\#T}$ coin 0=T
HHHT	$\frac{3}{4}\left(\frac{2}{3}\right)^3 \frac{1}{3}$	$\frac{1}{4}\left(\frac{1}{2}\right)^3 \frac{1}{2}$
HTHT	$\frac{3}{4}\left(\frac{2}{3}\right)^2 \left(\frac{1}{3}\right)^2$	$\frac{1}{4}\left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^2$
HHHT	$\frac{3}{4}\left(\frac{2}{3}\right)^3 \frac{1}{3}$	$\frac{1}{4}\left(\frac{1}{2}\right)^3 \frac{1}{2}$
HTTH	$\frac{3}{4}\left(\frac{2}{3}\right)^2 \left(\frac{1}{3}\right)^2$	$\frac{1}{4}\left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^2$

$$\alpha_1 = \frac{3}{4}$$

$$p_1 = \frac{2}{3}$$

$$q_1 = \frac{1}{2}$$

Step3: Compute Posterior



$$P(coin\ 0 = H \mid Seq) = \frac{P(coin0{=}H,\ Seq)}{P(coin0{=}H, Seq) + P(coin0{=}T, Seq)}$$

- Compute P(coin0, Seq)

	$\alpha_1 p_1^{\#H}(1-p_1)^{\#T}$	$(1-\alpha_1) q_1^{\#H}(1-q_1)^{\#T}$
Sequence	coin 0=H	coin 0=T
HHHT	0.074	0.0156
HTHT	0.037	0.0156
HHHT	0.074	0.0156
HTTH	0.037	0.0156

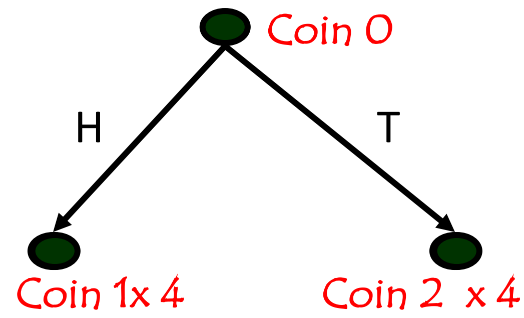
$$\alpha_1 = \frac{3}{4}$$

$$p_1 = \frac{2}{3}$$

$$q_1 = \frac{1}{2}$$

$$\frac{0.074}{0.074 + 0.0156} = 82.6\%$$

Step3: Compute Posterior



$$P(coin\ 0 = H \mid Seq) = \frac{P(coin0=H,\ Seq)}{P(coin0=H, Seq)+P(coin0=T, Seq)}$$

- Compute $P(coin\ 0 = H \mid Seq)$

	coin 0=H	coin 0=T
HHHT	82.6%	17.4%
HTHT	70.3%	29.7%
HHHT	82.6%	17.4%
HTTH	70.3%	29.7%

$$\alpha_1 = \frac{3}{4}$$

$$p_1 = \frac{2}{3}$$

$$q_1 = \frac{1}{2}$$

$$\frac{0.074}{0.074 + 0.0156} = 82.6\%$$

Step 2: Maximum Conditional Likelihood

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

- Now, compute the most likely value of the parameters. [recall the scenario I]

	coin 0=H	coin 0=T
HHHT	82.6%	17.4%
HTHT	*70.3%*	29.7%
HHHT	82.6%	17.4%
HTTH	*70.3%*	29.7%

Soft label assignment/ weighted instance

0.826 HHHHT
0.174 THHHT
0.703 HHTHT
0.297 THTHT
0.826 HHHHT
0.174 THHHT
0.703 HHTTH
0.297 THTTH

Step 2: Maximum Conditional Likelihood

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

- Now, compute the most likely value of the parameters. [recall the scenario I]

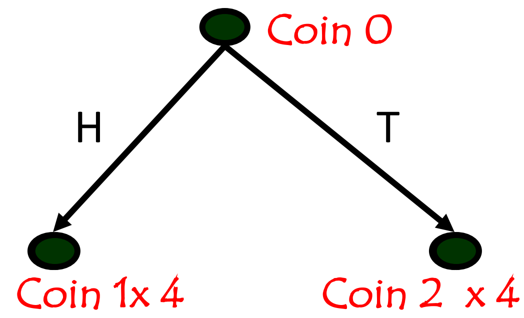
0.826 HHHHT
0.174 THHHT
0.703 HHTHT
0.297 THTHT
0.826 HHHHT
0.174 THHHT
0.703 HHTTH
0.297 THTTH

$$\alpha_2 = \frac{0.826\times 2 + 0.703\times 2}{4} = 0.765$$

$$p_2 = \frac{0.826\times 6 + 0.703\times 4}{0.826\times 8 + 0.703\times 8} = 0.635$$

$$q_2 = \frac{0.174\times 6 + 0.297\times 4}{0.174\times 8 + 0.297\times 8} = 0.592$$

Step3: Compute Posterior



$$P(coin\ 0 = H \mid Seq) = \frac{P(coin0=H,\ Seq)}{P(coin0=H, Seq)+P(coin0=T, Seq)}$$

❖ Compute P(coin0 | Seq)

$$\alpha_2 = 0.765$$

$$p_2 = 0.635$$

$$q_2 = 0.592$$

Sequence	coin 0=H	coin 0=T
HHHT		
HTHT		
HHHT		
HTTH		

Intuition of EM algorithm

- Use an iterative approach for estimating the parameters:
  - Guess the probability that a given data point came from Coin 1 or 2;   Generate fictional labels, weighted according to this probability.
  - Then, compute the most likely value of the parameters. [supervised learning]
  - Compute the likelihood of the data given this model.
  - Re-estimate the parameter setting: set them to maximize the likelihood of the data.

EM algorithm

- EM algorithm essentially solves $\mathrm{argmax}_{\Theta} \log \sum_Y P(X, Y \mid \Theta)$ by iteratively updating Θ (come back to this point later)
  - At iteration t, the model Θ^t
  - E-Step: Estimate $P(Y \mid X, \Theta^t)$
  - M-Step: Optimize

    $\max_{\Theta} \sum_{Y}[P(Y \mid X, \Theta^t) \log P(X, Y|\Theta)]$
- In general, it converges to a local maximum

Gaussian Mixture Models

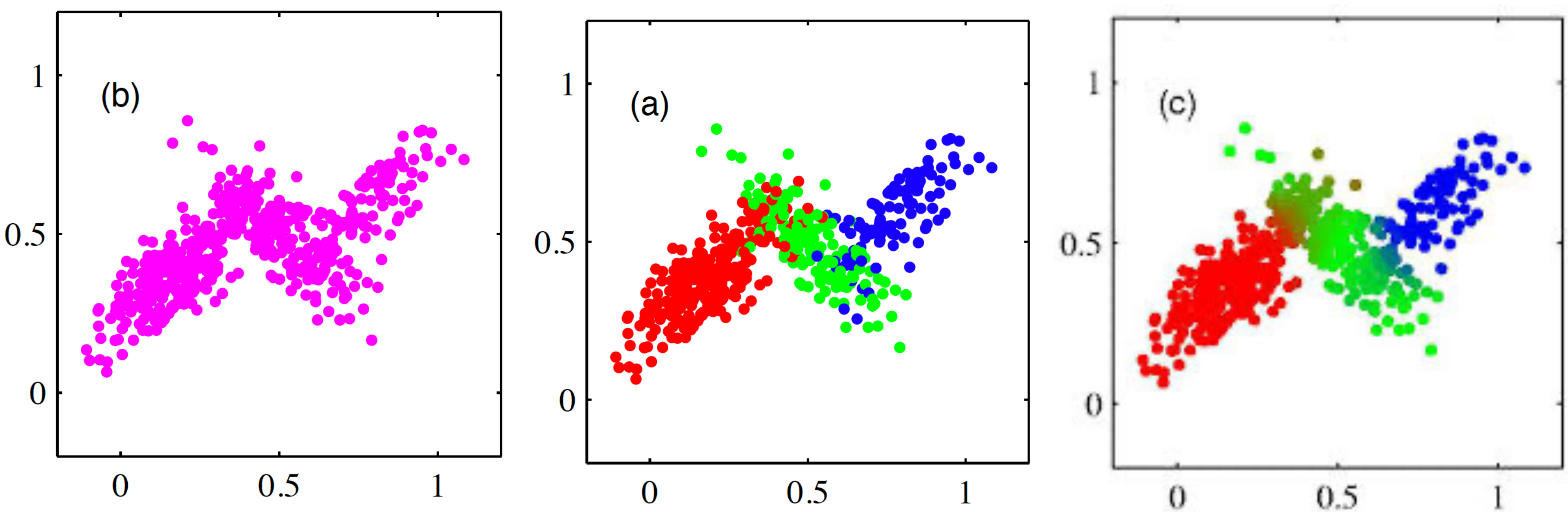
Recap: Gaussian mixture models

- Assume the probability density function for x as

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Hard cluster assignment

Soft cluster assignment



Iterative procedure (similar to k-means)

- Let θ represent all parameters $\{\omega_k, \mu_k, \Sigma_k\}$

Step 0: initialize $\boldsymbol{\theta}$ with some values (random or otherwise)

Step 1: compute γ_{nk} using the current $\boldsymbol{\theta}$

Step 2: update $\boldsymbol{\theta}$ using the just computed γ_{nk}

Step 3: go back to Step 1

E-step: Estimate γ_{nk}

- $\gamma_{nk} = P(z_n = k | x_n)$ the assignment of z_n to cluster k

e.g., probability x_n belongs to the red cluster, given the input and model

- posterior probability

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} = \frac{\overbrace{p(\boldsymbol{x}_n|z_n = k)}^{\boxed{N(x|\mu_k, \Sigma_k)}}\overbrace{p(z_n = k)}^{\boxed{\omega_k}}}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}$$

M-Step: Parameter estimation for GMMs

- If cluster assignments are observed $\{z_n\}$ are given
  - We know the cluster of each point
  - Let $\gamma_{nk} \in [0,1]$ is a soft assignment of instance n to cluster k,
- Then the maximum likelihood estimation is

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

- For ω_k: count the number of data points whose z_n is k and divide by the total number of data points (note that $\sum_k \sum_n \gamma_{nk} = N$)
- For $\boldsymbol{\mu}_k$: get all the data points whose z_n is k, compute their mean
- For $\boldsymbol{\Sigma}_k$: get all the data points whose z_n is k, compute their covariance matrix

GMM algorithm

- Step 0: randomly assign the cluster centers $\{\mu_k\}$ and covariance matrices $\{\Sigma_k\}$
- Step 1: Assign every point to the cluster based on posterior distribution $P(z_n = k|x_n)$

$$\boxed{p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}}$$

- Step 2: update the cluster centers $\{\mu_k\}$, and covariance matrices $\{\Sigma_k\}$ based on soft assignment $\gamma_{nk} = P(z_n = k|x_n)$

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

- Loop until it converges

γ_{nk} is similar to r_{nk} in K-means, but $\gamma_{nk} \in (0,1), r_{nk} \in \{0,1\}$

Generative v.s Discriminative Models

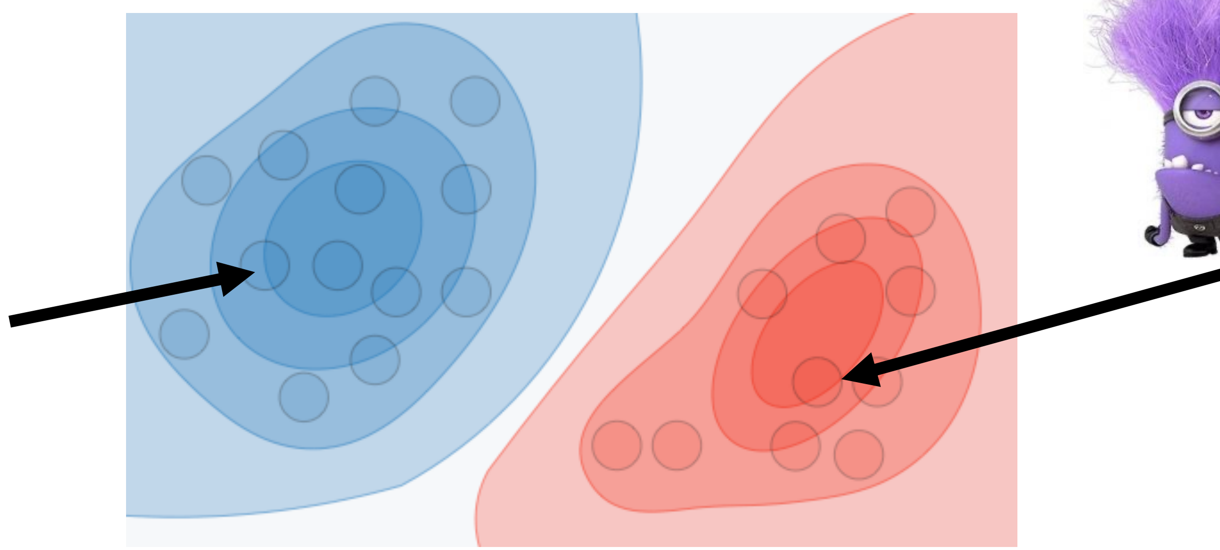
Example

V.S.

Generative (e.g., Naïve Bayes)

- Estimate $P(Y|X)$ through $P(X,Y)$
- MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i, x_i)$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

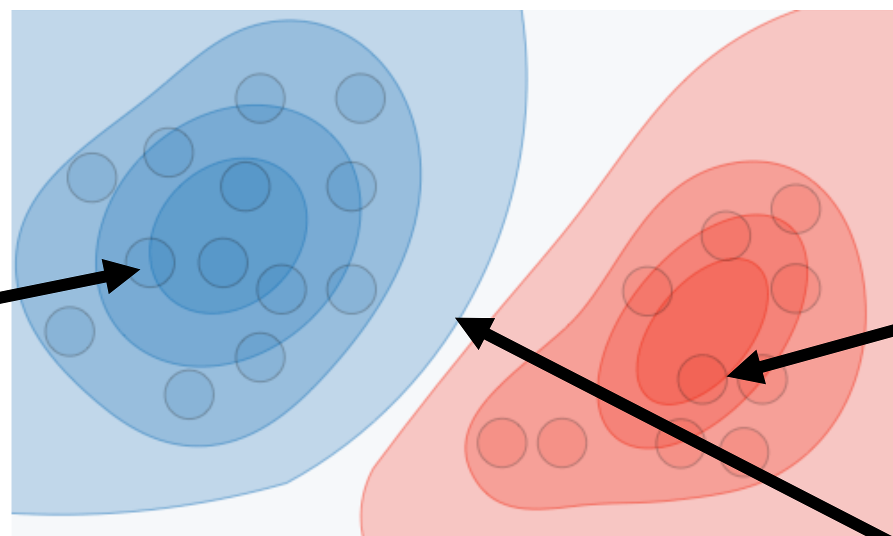


What is the input instances X look like if they belongs to the class Y

Generative (e.g., Naïve Bayes)

- Estimate $P(Y|X)$ through $P(X, Y)$
- MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i, x_i)$

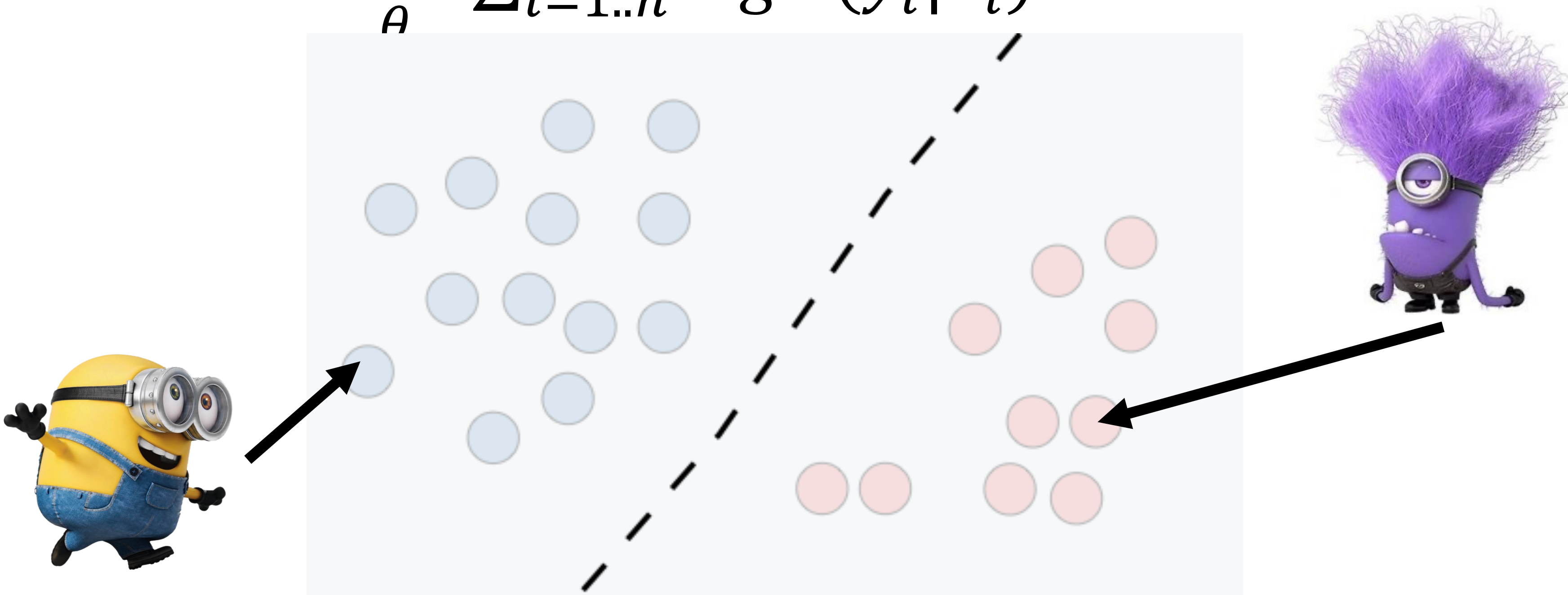
$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$



What is the input instances X look like if they belongs to the

Discriminative (e.g., logistic regression, SVM)

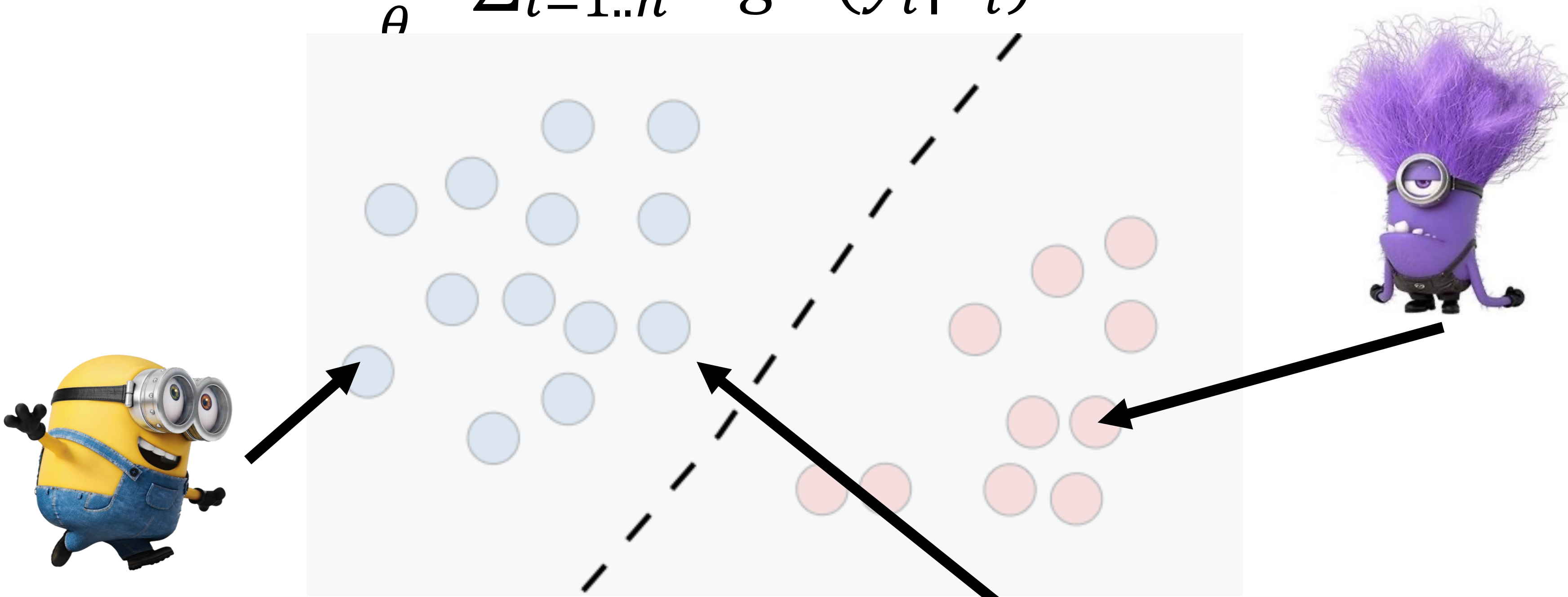
- Estimate $P(Y|X)$
- MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i|x_i)$



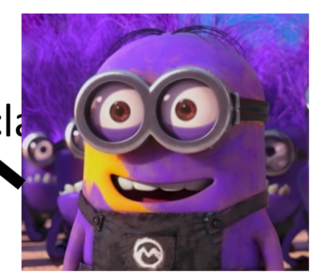
What features can be used to differentiate between different classes

Discriminative (e.g., logistic regression, SVM)

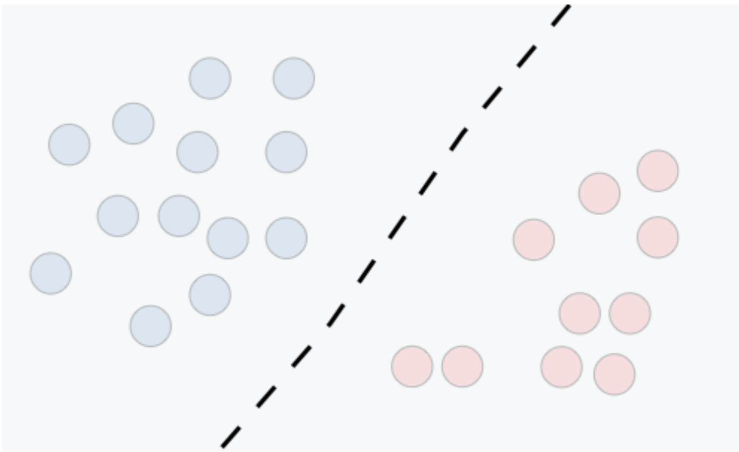
- Estimate $P(Y|X)$
- MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i|x_i)$



What features can be used to differentiate between different cl

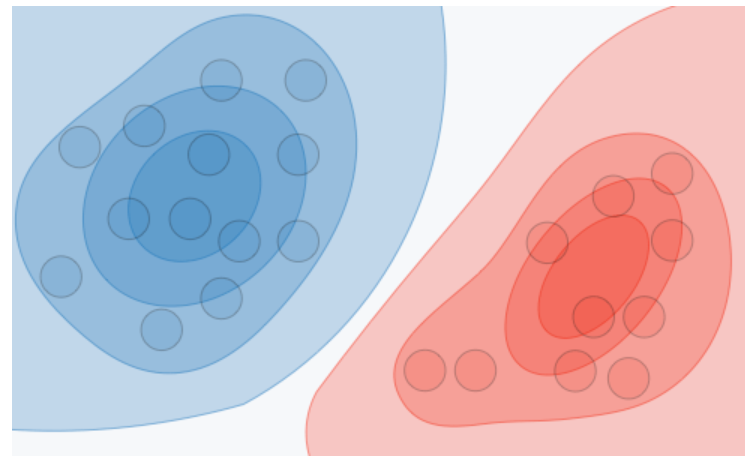


Discriminative vs Generative Models



Estimate $P(Y|X)$
MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i|x_i)$

Discriminative



Estimate $P(Y|X)$ through $P(X,Y)$
MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i, x_i)$

Generative

What features can be used to differentiate between different classes

A retrospective look at the course

Learning = generalization

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Tom Mitchell (1999)

We saw different “models”

what kind of a function should a learner learn

- K-NN
- Linear classifiers
- Decision trees
- Non-linear classifiers, feature transformations, neural networks

Different learning protocols

- Supervised learning
  - A *teacher* supplies a collection of examples with labels
  - The *learner* has to learn to label new examples using this data
- Unsupervised learning
  - No *teacher*, *learner* has only unlabeled examples

The theory of machine learning

Mathematically defining learning

- Entropy (information theorem)
- Probably Approximately Correct (PAC) Learning
- MLE, MAP
- SGD (optimization)

Some general recipes in ML

- Empirical Loss Minimization
  - Define loss and regularizer -> SGD
- Deep Learning
  - Define model architecture -> SGD
- Probabilistic models
  - Define model $P(Y|X)$ or $P(X,Y)$ -> MLE, MAP -> SGD, or closed form

Next Step

- If you're interested in ML/AI here are some relevant courses
  - CS145, CS148 – have significant overlap w/ CM146 but focus more on data.
  - CS161 (AI) – some overlap w/ cM146, but focuses on logic, reasoning, search
  - CS188 (NLP) – ML applications in NLP
  - Graduate-level courses – deep learning, NLP, Computer vision, probabilistic reasoning, etc…

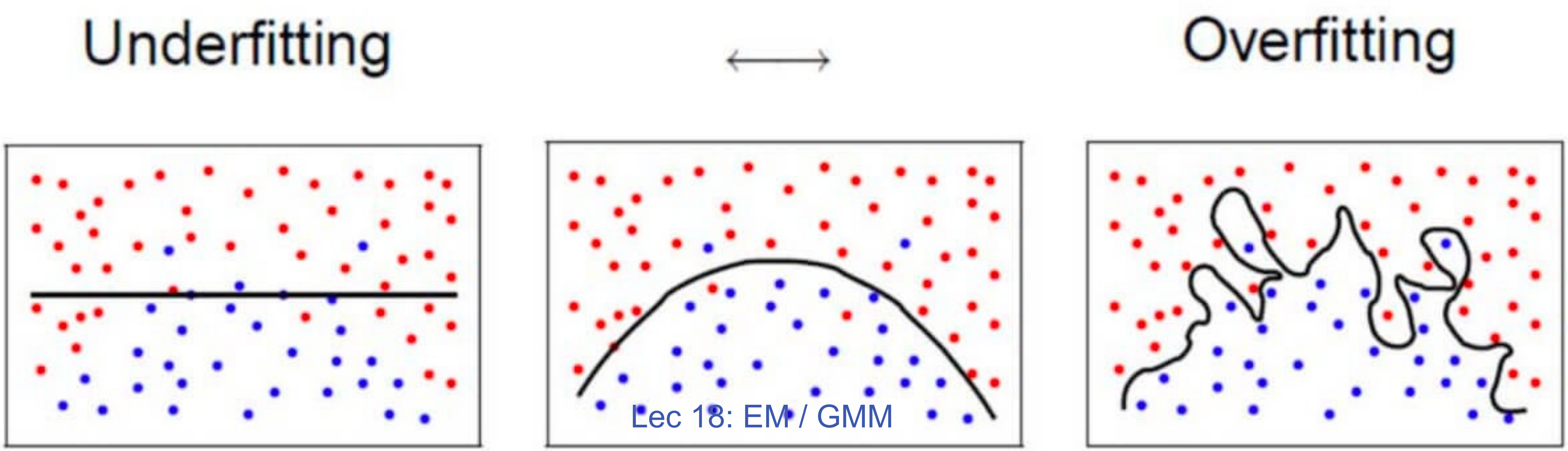
Practical advices

Bias and variance

Every learning algorithm requires assumptions about the hypothesis space.

Eg: "My hypothesis space is

- …linear"
- …decision trees with 5 nodes"
- …deep neural network with 12 layers"



Managing bias and variance

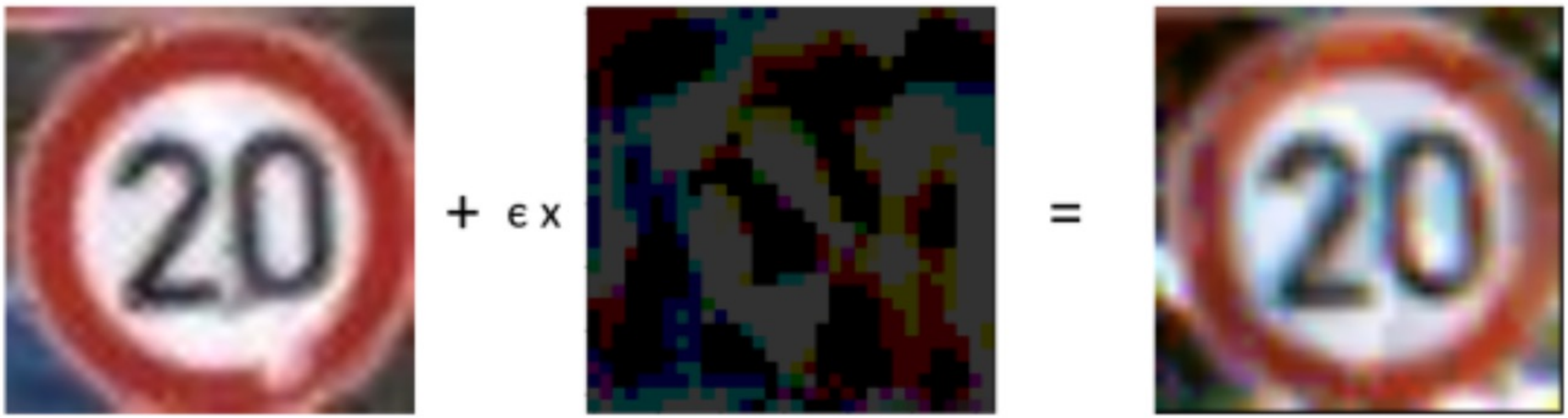
- Decision trees of a fixed depth
  - Increasing depth decreases bias, increases variance
- SVMs
  - Stronger regularization (i.e., smaller C in our formulation) increases bias, decreases variance
- K nearest neighbors
  - Increasing k generally increases bias, reduces variance
- Other approaches: Drop out, Ensemble, etc…

Tune your parameters!!

- Always tune parameters when comparing different settings / algorithms
- Never tune your parameters on the test set

ML is more than Curve Fitting

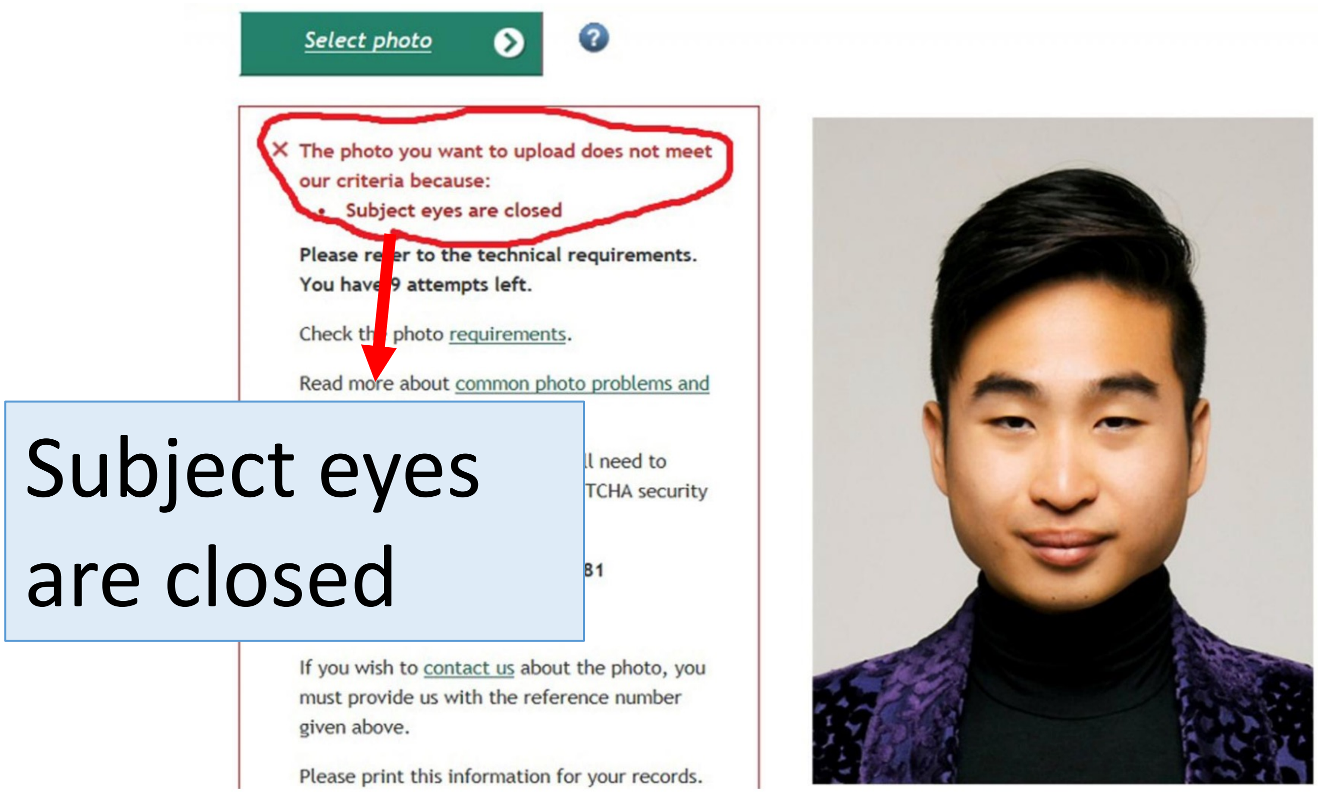
- Real world is adversarial



93%, 20 Km/h Sign | $sign(\nabla * J(\theta, x, y))$ | 90%, 80 Km/h Sign

ML is more than Curve Fitting

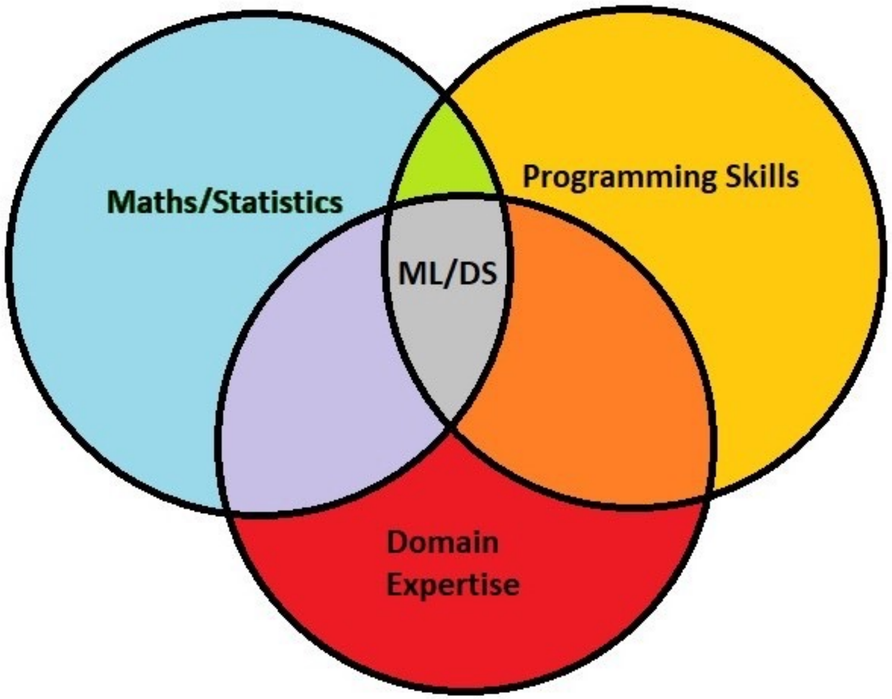
- Build system works well for everyone



A screenshot of New Zealand man Richard Lee's passport photo rejection notice, supplied to Reuters December 7, 2016. Richard Lee/Handout via REUTERS

ML is more than Curve Fitting

❖ Domain knowledge is important!



Thank you

- Please log in MyUCLA for course survey